PROJET 4 INFO

REALITE VIRTUELLE

Encadrants

Ronan Gaugne

Valérie Gouranton

Rapporteur

Jean-Louis Pazat

Étudiants

Edgar-Fernando Arriaga-Garcia

Charles-Henri Babiaud

Clément Grellier

Quentin Petit

Jérôme Ricœur

Florent Violleau

Rapport final

31 mai 2012

INSA RENNES

Table des matières

Remerciements **3**

Introduction **5**

1 Avancement du projet **8**
1.1 Cahier des charges 8
1.2 Spécification 8
1.2.1 Rappel des objectifs 8
1.2.2 Objectifs de spécification accomplis 9
1.3 Implémentation 12
1.3.1 Rappel des objectifs de conception 12
1.3.2 Implémentation réelle 13
1.4 Phases de tests réalisées 14
1.4.1 Tests unitaires 14
1.4.2 Tests utilisateur 14
1.5 Difficultés rencontrées 14

2 Planification **16**
2.1 Planification initiale 16
2.1.1 Prévision initiale 16
2.1.2 Tâches initiales 16
2.1.3 Chronologie initiale 16
2.1.4 Diagramme de Gantt initial 17
2.2 Planification réelle 19
2.2.1 Présentation de la démarche 19
2.2.2 Jalons 19
2.2.3 Diagramme de Gantt 20
2.2.4 Retards et causes 22
2.2.5 Chiffres clés 22
2.3 Bilan sur les paramètres et la démarche de planification 23
2.3.1 Bilan sur les paramètres de planification 23
2.3.2 Bilan sur les méthodes de gestion de projet 23

3 Perspectives et méthodologie de la reprise du projet **25**
3.1 Travail restant et perspectives de reprise du projet 25
3.2 Vers de nouveaux horizons 25
3.2.1 Choix des technologies 25
3.2.2 Reprise de notre travail 26
3.3 Répartition de la salle 26

Conclusion **27**

Bibliographie **28**

Remerciements

Tout au long de notre projet de l'année, nous avons été aidés, appuyés, conseillés par nos encadreurs. Ils ont su faire preuve de patience ainsi que de disponibilité. C'est donc à ce titre que nous adressons en premier lieu nos remerciements à Mme Valérie GOURANTON et M. Ronan GAUGNE, nos encadreurs. Notre gratitude va également à notre rapporteur M. Jean-Louis PAZAT qui a pris le soin et le temps d'examiner avec attention nos rapports, d'assister à nos soutenances. Nous remercions également M. Eric ANQUETIL, qui a été présent lors de toutes nos soutenances et qui nous aura permis d'avoir une vision extérieure du projet.

Ce projet n'aurait pas pu voir le jour sans l'aide financière conséquente du département informatique accordée par M. Ivan LEPLUMEY pour ce projet.

Nous tenons à exprimer toute notre reconnaissance à M. Thierry COZIC et M. Jean-Luc NEVEU pour le support technique et le bon fonctionnement du matériel. Nous adressons particulièrement toute notre gratitude à M. Bertrand MÉNAGE qui a été d'une disponibilité exemplaire, d'une grande efficacité et d'un grand soutient quant à l'installation et l'utilisation du matériel dans la salle μRV.

Nous tenons également à remercier chaleureusement les chercheurs de la salle Immersia à savoir Florian Nouviale pour le support technique sur OpenMASK et particulièrement Alain Chauffaut qui a pris sur son temps personnel pour alimenter le wiki d'OpenMASK et étoffer la documentation existante, pour avoir fait des présentations sur le fonctionnement d'OpenMASK et pour avoir apporté une aide précieuse pour la construction de la partie OpenMASK de notre application. Enfin, toutes les personnes qui s'occupent de la salle Immersia qui nous ont permis de tester des applications dans une grande salle de réalitée virtuelle.

En dernier lieu, nous remercions Sébastien Kuntz, fondateur d'« i'm in VR » et de l'association VR Geek, qui a fait une conférence sur les techniques de réalitée virtuelle et fourni et installé son logiciel MiddleVR sur les machines de la salle μRV. Grâce à sa documentation, nous avons pu créer un module qui permet de récupérer les évènements de la Wiimote. Enfin, il a aussi fourni de précieux conseils quand aux choix du matériel à acheter pour la salle μRV.

Introduction

L'objectif de notre projet était de réaliser une application permettant l'interaction entre deux utilisateurs. Elle avait aussi pour objectif de réutiliser le projet développé par la promotion précédente[3], à savoir une application de construction de Lego.

C'est chose faite et réussie. En effet, nous avons atteint 90% de nos objectifs fixés en début d'année. Le premier utilisateur, à savoir un coureur se déplaçant dans un monde, peut se mouvoir librement dans un labyrinthe en utilisant la reconnaissance du mouvement par la Kinect. Notre logiciel gère bien entendu les collisions entre le coureur et les différentes briques Lego du labyrinthe car sinon l'intérêt en serait fortement diminué. Cette fonctionnalité a été très peu évoquée dans les rapports précédents mais s'est avérée primordiale. Pour l'autre utilisateur, le constructeur, il peut déplacer son point de vue librement en utilisant la Wiimote et poser des pièces dans le labyrinthe pour empêcher le coureur d'atteindre la sortie.

Ainsi, après avoir effectué les différentes phases d'analyse tout au long de l'année, ce rapport a pour but de s'intéresser aux évolutions de notre projet et de faire un retour constructif et critique sur nos rapports précédents. Nous allons exposer les modifications qui ont pu être établies tout au long du projet et notamment lors du développement. Un retour sur la planification s'impose également pour connaître les évolutions effectuées quant à la répartition des ressources.

Dans un premier temps, nous revenons sur différents rapports, à savoir : spécification, conception et planification. Ainsi, nous faisons l'état de l'avancement du projet. Nous reprenons le cahier des charges établi au commencement du projet et nous faisons un bilan sur les rectification des différentes phases du projet. Nous exposons aussi la manière dont nous avons implémenté Block3D et les différentes phases de tests réalisés. Nous revenons dans une deuxième partie sur la planification du projet Block3D en faisant une comparaison directe entre la planification initiale et la planification réelle. Enfin, en dernier lieu, nous présentons les différentes perspectives et méthodologies quant à la reprise du projet vis à vis des années suivantes. Ainsi, dans cette section, nous faisons un état du travail restant et des nouvelles fonctionnalités qui pourraient être implémentées. Nous donnons aussi des conseils et astuces qui ont permis une bonne gestion de la salle et du projet.

Vous trouverez en annexe de ce rapport, des liens vers les manuels d'installation et d'utilisation de Block3D.

Collaboration avec les autres projets

Il est important de parler de la collaboration avec d'autres projets car c'est une partie inhérante à notre projet. En effet, un groupe de $3^{ème}$ année à travaillé avec nous et s'est occupé de la partie génération de labyrinthe. Ce projet

d'étude pratique s'appelle Daedalus (cf. figure 1). Ils ont implémenté un algorithme de génération de labyrinthe. Lors de la génération du terrain, on peut choisir de multiples paramètres comme la taille (largeur, longueur) et la méthode de construction. Ils ont par ailleurs réalisé une interface Qt en C++ pour paramétrer plus facilement les différentes options de génération. En accord avec les membres de notre projet, le projet Daedalus a ajouté une fonctionnalité : la gestion de la couleur des briques. Cette fonctionnalité a permis par la suite de proposer lors de la génération de la carte un dégradé de couleur pour offrir une aide au coureur (les briques deviennent plus claires quand elles sont proches de l'objectif cf. figure 2). Par ailleurs, nous pouvons voir que les $3^{ème}$ années ont aussi implémenté une technique d'optimisation pour augmenter les FPS [1] (cf. figure 2) car certaines briques sont déformées volontairement.

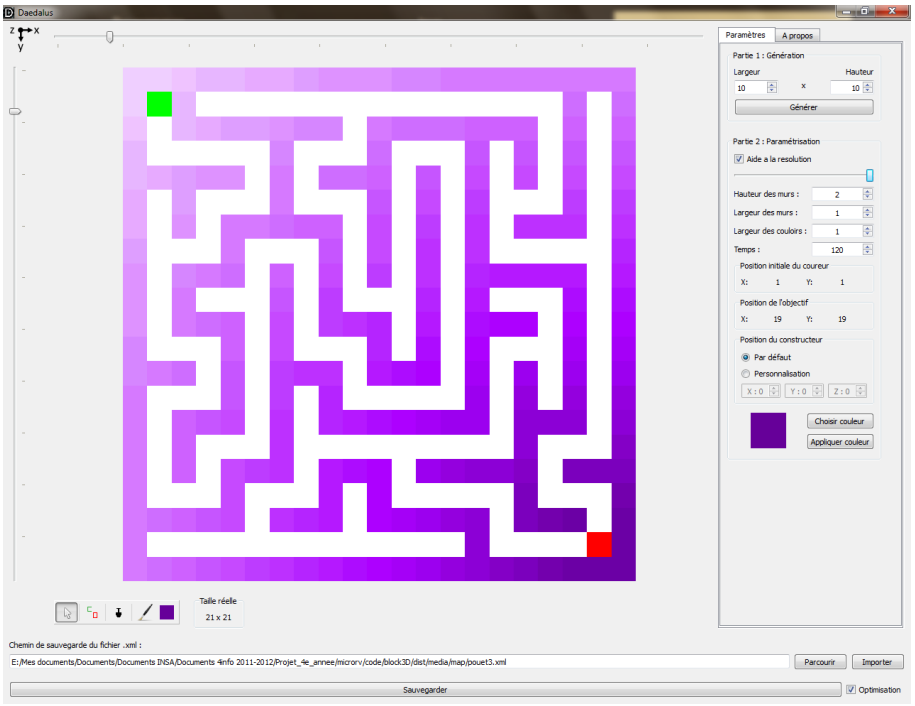

FIGURE 1 – Interface du projet du groupe de $3^{ème}$ année

1. FPS : Frames Per Second.

FIGURE 2 – Labyrinthe généré par le projet du groupe de $3^{ème}$ année

Concernant le groupe de $5^{ème}$ année, la période dédiée à leur projet s'est terminée en février. Ils ont étendu les fichiers de configuration OpenMASK « .omk » que les stagiaires avaient réalisés sur L3G0 cet été pour permettre une collaboration. Néanmoins, ils nous ont fait part de leurs difficultés à maîtriser OpenMASK notamment à cause de son manque de documentation. Ils ont en parti remédié à ce manque en fournissant une documentation pour compiler et installer les projets pour l'installation des exemples de la grande salle Immersia. Nous avons donc pris en compte dans notre conception cette difficulté en limitant dans un premier temps les interactions avec OpenMASK au strict nécessaire puis en ajoutant au fur et à mesure de nouvelles fonctionnalités.

1 Avancement du projet

Dans cette première partie, nous décrivons l'avancement du projet suivant le cahier des charges et les objectifs que nous nous étions fixés. Les choix technologiques ont pu être modifiés et leur évolution est abordée. Les tests réalisés ainsi que les difficultés rencontrées sont également traités.

1.1 Cahier des charges

Le cahier des charges lié à notre projet comportait deux éléments notables.

Le premier élément est l'administration de la salle μRV. Nous sommes, en effet, les référents techniques de la salle et les seuls étudiants à en posséder les clés. Chaque groupe de projet ayant besoin de la salle devait passer par nous. Nous avons donc assumé ce rôle tout au long de l'année. Nous avons également dû être présents lors des journées portes-ouvertes pour présenter la salle μRV.

Le deuxième élément du cahier des charges est la réalisation d'une application phare mettant en avant la salle de réalité virtuelle. Nous étions libre du choix de l'application, notre application satisfait donc le cahier des charges.

Les parties suivantes sont consacrées à l'avancement de notre application par rapport aux spécifications que nous avions nous-mêmes définies.

1.2 Spécification

Au niveau de la spécification, la plupart de nos objectifs ont été remplis. Nous allons rappeler dans un premier temps les objectifs de spécification que nous nous étions fixés puis voir les différences avec notre application réelle.

1.2.1 Rappel des objectifs

Deux utilisateurs devaient pouvoir évoluer dans un labyrinthe de briques Legos.

Le premier utilisateur, dit coureur, est plongé au cœur du labyrinthe et doit trouver un objectif en un temps limité. Il est représenté sur l'écran par un avatar dirigé par les mouvements de l'utilisateur grâce à une Kinect. Il dispose de lunettes Nvidia[17] pour percevoir l'environnement en 3D.

Le deuxième utilisateur, dit constructeur, a une vue omnisciente du labyrinthe. Il peut se déplacer et poser des pièces temporaires limitées en nombre et en temps pour freiner le coureur. Il dispose d'une WiiMote et d'un Nunchuck pour effectuer ses actions et perçoit l'environnement au travers de lunettes immersives Vuzix Wrap 920[18].

Un menu peut être invoqué pour mettre le jeu en pause. Des fonctions sont alors disponibles pour quitter l'application, afficher l'aide ou reprendre la partie.

Un administrateur était également prévu pour lancer l'application et régler les paramètres de celle-ci.

1.2.2 Objectifs de spécification accomplis

Nous nous intéressons dans cette partie aux fonctionnalités que nous avions prévu.

Fonctionnalités du coureur

Les fonctionnalités liées au coureur ont presque toutes été respectées.

Les actions de translation à gauche et à droite n'ont pas été implantées par soucis de temps. Malgré le fait que le codage soit minime, la création d'animation correspondante pour l'avatar prend beaucoup de temps. Nous avons préféré privilégier d'autres tâches plus importantes.

Le saut étant optionnel, nous avons également décider de l'abandonner au vu des tâches restantes.

Fonctionnalité	Description	Niveau d'implémentation
Avancer	Avance l'avatar dans le labyrinthe en ligne droite	✔
Reculer	Recule l'avatar	✔
Tourner à gauche	Tourne l'avatar vers la gauche	✔
Tourner à droite	Tourne l'avatar vers la droite	✔
Translater à gauche	Déplace l'avatar vers la gauche sans changer la direction de la caméra	✗
Translater à droite	Déplace l'avatar vers la droite sans changer la direction de la caméra	✗
Sauter	Fonction facultative pour notre projet, mais pourrait permettre à l'avatar de sauter au-dessus de certains obstacles bas	✗
Accéder au menu	Affiche le menu et met le jeu en pause	✔

FIGURE 3 – Vue du coureur

Fonctionnalités du constructeur

Les fonctionnalités du constructeur, quant à elles, ont toutes été implantées.

Fonctionnalités	Description	Niveau d'implémentation
Zoomer	Rapproche la vue de l'environnement	✔
Dézoomer	Recule la vue	✔
Translater la vue	Translate la vue	✔
Orienter la vue	Oriente la vue en gardant la même position	✔
Ajouter une pièce	Ajoute une pièce temporairement dans l'environnement	✔
Changer la pièce courante	Sélectionne la pièce susceptible d'être ajouté	✔
Accéder au menu	Affiche le menu et met le jeu en pause	✔

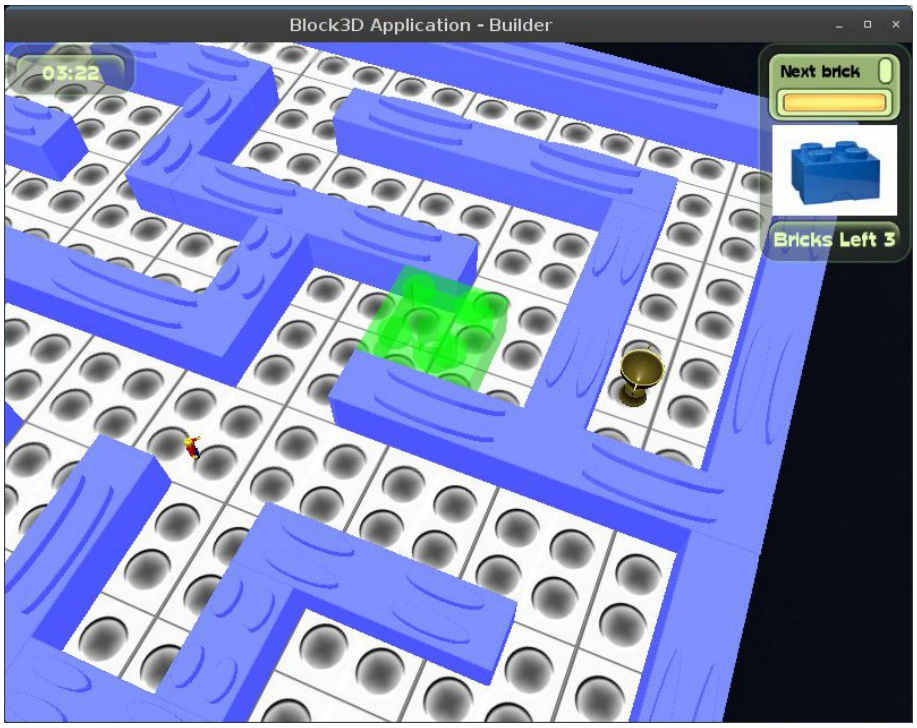

FIGURE 4 – Vue du constructeur

Fonctionnalités du menu

Le menu, pouvant être appelé par le coureur ou le constructeur, a vu de nouvelles fonctionnalités apparaître. On peut en effet choisir le labyrinthe et régler les paramètres sonores.

Fonctionnalités	Description	Niveau d'implémentation
Reprendre la partie	Reprend la partie en sortant de la pause	✔
Choisir le labyrinthe	Permet de choisir le labyrinthe dans lequel évolue le coureur	✔
Choisir les paramètres sonores	Permet de choisir les paramètres sonores de l'application	✔
Aide	Affiche l'aide	✔
Quitter le jeu	Quitte le jeu	✔

FIGURE 5 – Affichage du menu

Fonctionnalités de l'administrateur

L'administrateur correspond en réalité à un fichier de configuration où l'on peut choisir certains paramètres. Une configuration plus manuelle permet également de déterminer les labyrinthes disponibles, les musiques disponibles, etc.

Il est important de noter qu'un éditeur de labyrinthe a été mis au point par le groupe de 3INFO Daedalus. Il est donc facile de réaliser un labyrinthe et de l'utiliser ensuite en jeu.

Les objectifs que nous avions spécifiés ont donc quasiment tous été atteints.

1.3 Implémentation

La conception a quelque peu été modifiée au fur et à mesure de l'évolution du projet. Nous abordons dans cette partie les changements apportés.

1.3.1 Rappel des objectifs de conception

Rappelons dans un premier temps les composantes logicielles que nous comptions utiliser ainsi que les paquetages prévus.

Composantes logicielles

Les logiciels initialement prévus pour être utilisés avec notre application sont :
- Ogre3D [2] : le moteur 3D de notre application.

2. OGRE[4] : Object-Oriented Graphics Rendering Engine

- OgreAL [3] : permet d'utiliser du son avec Ogre3D. OgreAL utilise OpenAL [4].
- VRPN [5] : permet de récupérer les informations de périphériques (Wiimote, Kinect, etc).
- OpenMASK [6] : permet la réalisation d'une application distribuée. Il utilise MPI [7].

Paquetages

Différents paquetages avaient été prévus lors de la conception de notre application :

- le paquetage « model » fournit une abstraction des données sur l'univers virtuel. Il possède les sous-paquetages suivants :
  - « data » : permet de stocker les différents types de données,
  - « parser » : sert à traiter les fichiers de terrains,
  - « time » : gère le temps,
  - « element » : regroupe l'ensemble des données du monde virtuel ;
- le paquetage « interaction » traite les données brutes des périphériques pour les mettre en forme. Il possède les sous-paquetages suivants :
  - « wiimote » : récupère les informations de la Wiimote,
  - « kinect » : récupère les informations de la Kinect ;
- le paquetage « plugin » représente les plugins OpenMASK ;
- le paquetage « ogre » s'occupe de la partie graphique de l'application ;
- le paquetage « OMK » montre les dépendances des classes d'OpenMASK utilisées ;
- le paquetage « test » contient l'ensemble des tests unitaires et d'intégrations de l'application.

1.3.2 Implémentation réelle

L'implémentation réelle a quelque peu différé par rapport à la conception prévue.

Composantes logicielles

Il est apparu que Ogre3D ne permettait pas de détection de collision nativement. Nous avons donc dû utiliser un détecteur de collision en plus de Ogre3D. Nous avons choisi NxOgre[7] lequel est un wrappeur du moteur de physiques de nVidia appellé PhysX [9] et qui permet d'utiliser ce dernier avec plusieurs systèmes de rendu. Pour arriver a utiliser NxOgre avec Ogre nous avons dû utiliser Critter[8]. Celui-ci rend une interface entre Ogre et NxOgre et rend aussi un débogueur visuel qui nous a beaucoup aidé pour la mise en oeuvre de la détection des collisions. L'implémentation du moteur physique est une modification qui donne beaucoup de possibilités pour des futures modifications car il possède

3. OgreAL[5] : Object-Oriented Graphics Rendering Engine Audio Library
4. OpenAL[6] : Open Audio Library, bibliothèque fournissant une interface de programmation audio 3D
5. VRPN[11] : Virtual Reality Peripheral Network
6. OpenMASK[13] : Modular Animation and Simulation Kit
7. MPI[14] : Message Passing Interface

des fonctionnalités comme la gestion des squelettes, des fluides, des vêtements, de la gravité, des dynamiques des véhicules, des corps souples et rigides, des champs de force et serialization des données. Avec toutes les fonctionnalités que nous offre PhysX, il est possible d'implémenter des fonctionnalités pour réaliser un jeu de plateforme. Par exemple, des fonctionnalités qui n'ont été ajoutées mais qui est possible est la gestion des sauts et l'ajout des champs de force pour faire plus d'interaction avec l'utilisateur. Au niveau de la programmation, la gestion des collisions a posé des difficultés, notamment parce qu'il n'y avait pas de correspondances entre le système de coordonnées de NxOgre et celui d'Ogre.

Par rapport à NxOgre la version utilisée est la version « BuggySwires » et la version de PhysX est la 2.8.4. Malheureusement même si PhysX est multiplateforme, NxOgre ne fonctionne que sous Windows. Nous avons donc eu recours à de la compilation conditionnelle pour pouvoir avoir une version sous Linux, mais sans détection de collision.

Concernant VRPN, un client a été réalisé pour la Wiimote, mais nous avons utilisé le logiciel FAAST[12] pour gérer la Kinect. En effet, il intègre un serveur et un client VRPN transparent à l'utilisateur, ce qui simplifie grandement l'utilisation.

Enfin, la distribution sur plusieurs machines à l'aide de OpenMASK n'est pas tout à fait fonctionnelle. Une ébauche a été réalisée mais OpenMASK est dur à utiliser et est peu documenté.

Paquetages

Peu de modifications ont eu lieu au niveau des paquetages.

Un nouveau sous-paquetage est apparu dans le paquetage « model ». Il s'agit du sous-paquetage « sound ». Notre application possède en effet des effets sonores ainsi qu'une musique d'ambiance.

Les paquetages « plugins » et « OMK » ont quant à eux disparu. Ils ont été remplacés par une extension OMK avec des fichiers de configuration.

1.4 Phases de tests réalisées

Tout le long du projet des tests ont été effectués, notamment des tests unitaires et des tests utilisateur. Cela a permis de conserver un projet cohérent et fiable tout au long du développement de l'application.

1.4.1 Tests unitaires

Des tests unitaires ont été réalisés à l'aide des bibliothèques CppUnit[15] et CxxTest[16] sur les fonctions du modèle de notre application. Cela a permis un débogage aisé de ces fonctions et d'assurer une base solide à notre application.

1.4.2 Tests utilisateur

Des tests utilisateurs ont pu être réalisés auprès du groupe de 3INFO Daedalus. Nous avons pu avoir un premier retour utilisateur permettant la correction et l'amélioration de notre application.

1.5 Difficultés rencontrées

Un certain nombre de difficultés ont pu être rencontrées lors du développement de notre application.

Une grande partie de ces difficultés était liée à l'utilisation de nombreux logiciels extérieurs : Ogre, OgreAL, OpenAL, NxOgre, Critter, PhysX, VRPN, OpenNI[10], OpenMASK, etc. En effet, chacun d'eux doit être installé et fonctionner correctement pour le bon fonctionnement de notre application.

Cependant, la principale difficulté que nous ayons pu rencontrer est sans doute OpenMASK. Sa prise en main difficile et son manque de documentation en font un logiciel difficile à maîtriser. De plus, un certain nombre de problèmes est survenu lors de l'installation d'OpenMASK et de la communication entre deux machines. Malgré l'aide de personnes travaillant à l'IRISA/Inria, l'utilisation d'OpenMASK est donc restée ardue.

Des problèmes techniques sont également survenus :
- Les lunettes Wrap 920 sont tombées en panne en cours d'année.
- Les Wiimotes ont dû être changées ainsi que leurs nunchuks au profit de leurs versions officielles.
- L'ordinateur utilisé dans la salle μRV ne permet pas le branchement du vidéoprojecteur et des lunettes Wrap 920 et même temps (nécessité d'avoir deux port VGA).

Enfin, deux ordinateurs sur les trois mis à notre disposition souffraient d'un cruel manque de performance. Au final, un seul ordinateur nous a réellement été utile, les deux autres ne pouvant servir qu'à des tâches minimes.

Tous ces problèmes nous ont donc freiné dans l'avancement de notre application.

2 Planification

La planification du projet est un aspect important du projet. Elle permet d'évaluer quelle a été la planification initiale puis les différences qu'il y a eu vis à vis de cette planification. Il s'agit alors d'évaluer quels ont été les points où la prévision n'a pas été cohérente, et ceux où la prévision s'est révélée satisfaisante.

2.1 Planification initiale

Il est fait état dans cette partie, de rappels et de précisions concernant la planification initiale : la prévision initiale, et chiffres initiaux, les principales tâches planifiées dans la planification initiale, et l'aspect chronologique de cette première planification. La planification initiale est présentée d'une manière plus concise dans le précédent rapport de planification mais les principaux éléments de la planification initiale sont tout de même présentés.

2.1.1 Prévision initiale

Dans notre prévision initiale, nous avons abordés les sujets suivants, avec une solution ou un découpage bien précis :

- Un **découpage en tâches** et en sous-tâches des processus de développement, conception, test, présentation et de livraison.
- Une **répartition de l'équipe** en 3 équipes de 4 personnes, respectivement les équipes « Coureur », « Constructeur » et « Matériel ».
- L'établissement de **4 jalons**.
- Des **méthodes de gestions de projet** : méthode SCRUM intégrée dans un processus unifié.
- L'établissement du **périmètre de notre projet** (maintenabilité, etc.).
- Une charge de **825 heures** initialement planifiée pour le $2^{ème}$ semestre.

Cette planification initiale a été réalisée fin janvier. Sa livraison a été fixée au démarrage de la tâche de réalisation du rapport de conception.

2.1.2 Tâches initiales

Les tâches initialement planifiées sont :

- La **conception** : livraison du rapport
- La **gestion des entrées-sorties** : événements de la Kinect, de la Wiimote, corrélation lunettes-application, intégration de Ogre3D dans OpenMASK, et collaboration OpenMASK
- Le **modèle** : macro-tâches coureurs et constructeur, ainsi que gestion du son et du temps
- La **gestion graphique**
- L'**administration** de l'application
- L'**intégration** : projets de $4^{ème}$ et $5^{ème}$ année, tests et recette.
- La **livraison** : rapport, documentation et présentation

2.1.3 Chronologie initiale

La chronologie initiale est présentée à la figure 6.

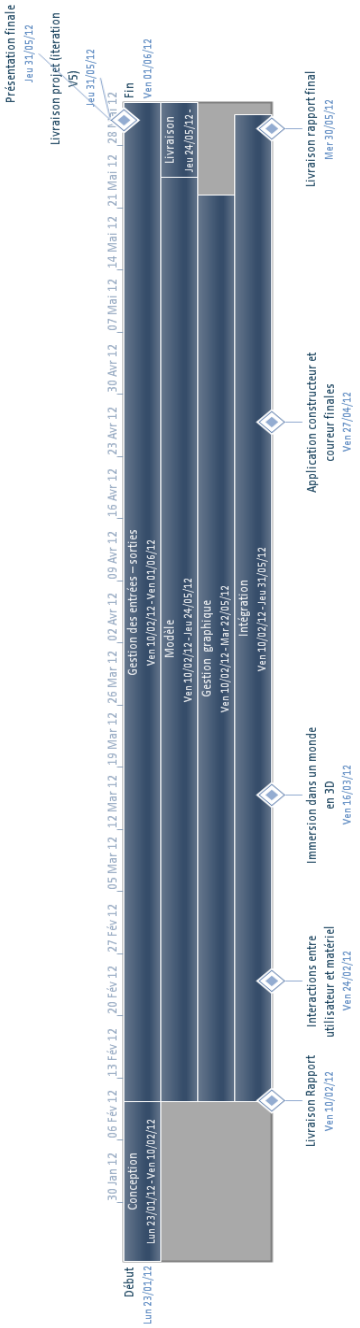

FIGURE 6 – Chronologie initiale

2.1.4 Diagramme de Gantt initial

Le diagramme de Gantt initial est à présenté à la figure 7

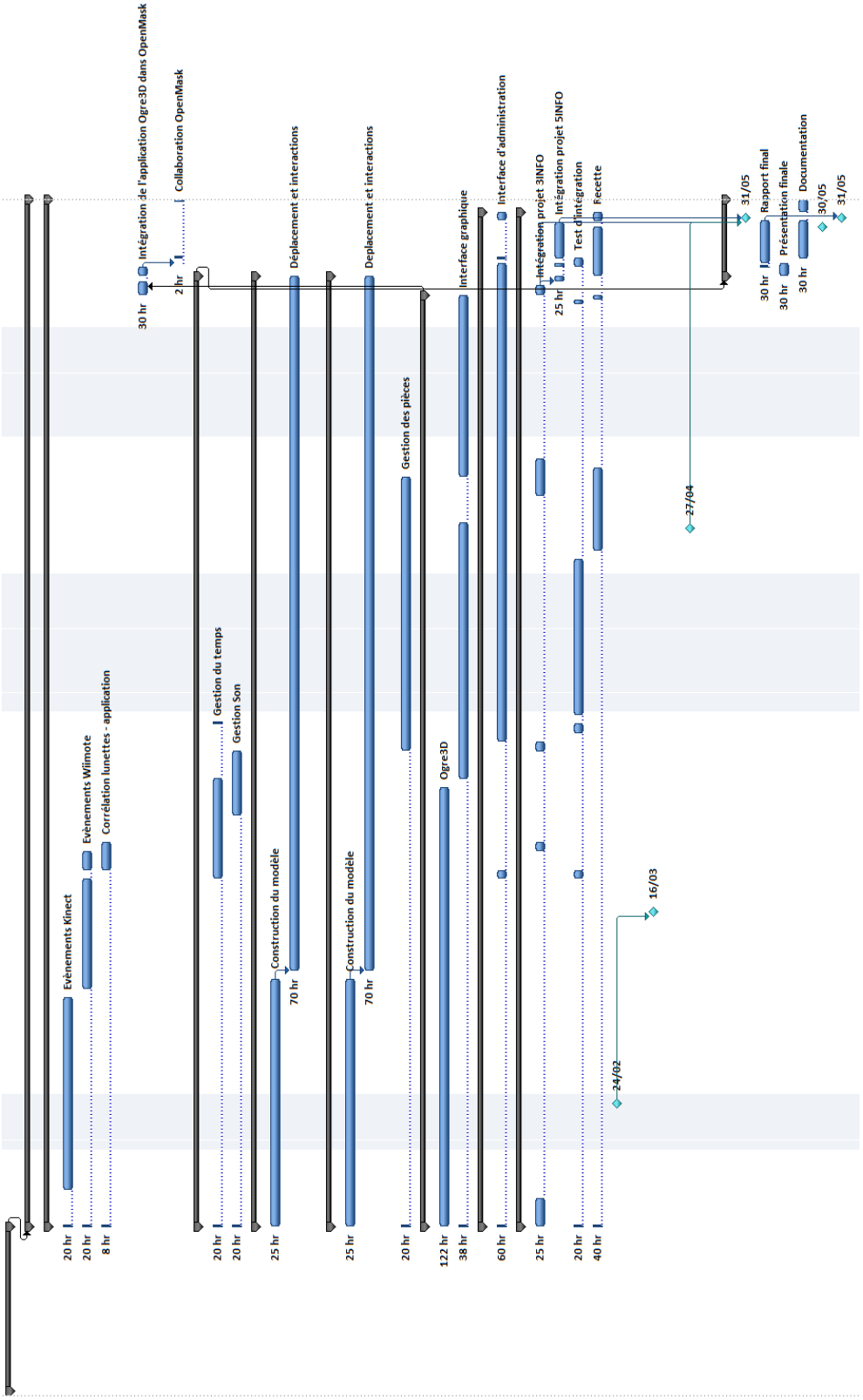

FIGURE 7 – Diagramme de Gantt initial

2.2 Planification réelle

L'analyse de la planification est étudiée : la **démarche** effectuée, les nouveaux **jalons** ainsi que le **diagramme de Gantt** associé. Enfin, nous faisons une brève **analyse des causes** expliquant les retards sur certaines tâches et indiquons quelques **éléments quantitatifs** permettant de qualifier notre projet.

2.2.1 Présentation de la démarche

Cette analyse s'appuie sur la participation au projet et sur plusieurs documents, à savoir :

- Le rapport de planification initiale
- Un suivi de planification à mi-parcours de développement (fin mars)
- Des indications du dépôt SVN et de la forge
- Les comptes-rendus de réunion
- Le suivi des heures
- L'état actuel de l'application

C'est sur cet ensemble que nous établissons une analyse de la planification réelle. **Nous nous appuierons tout particulièrement sur les différences entre la planification initiale, la planification à mi-parcours et la planification finale**.

2.2.2 Jalons

Plusieurs jalons ont été fixés initialement, mais suite au retard pris par certaines tâches, nous avons dû peu à peu reculer certains jalons. Ainsi, la figure 8 donne un aperçu de la date donnée aux jalons, en fonction de l'avancement du projet. En rouge apparaissent les jalons initiaux, en orange les dates des jalons de mi-parcours, et en rouge les jalons réels. Remarquons que pratiquement un mois sépare le dernier jalon prévisionnel et réel.

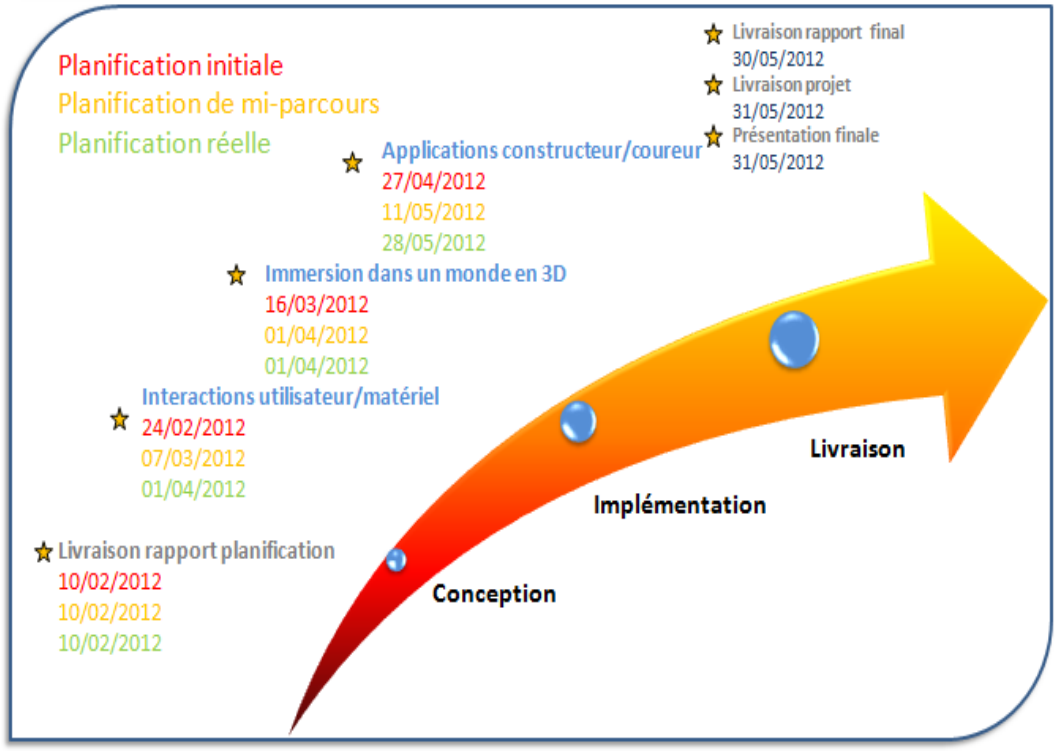

FIGURE 8 – Jalons initiaux et réels

2.2.3 Diagramme de Gantt

Le diagramme de Gantt final permet, avec plus de précision, de savoir quel a été notre avancement dans les tâches à la date de livraison du projet. Il est situé figure 9. On observe également que certaines tâches, mises en valeur sur le schéma, par un trait non-présent de gauche à droite de la tâche, n'ont pas été complètement réalisées.

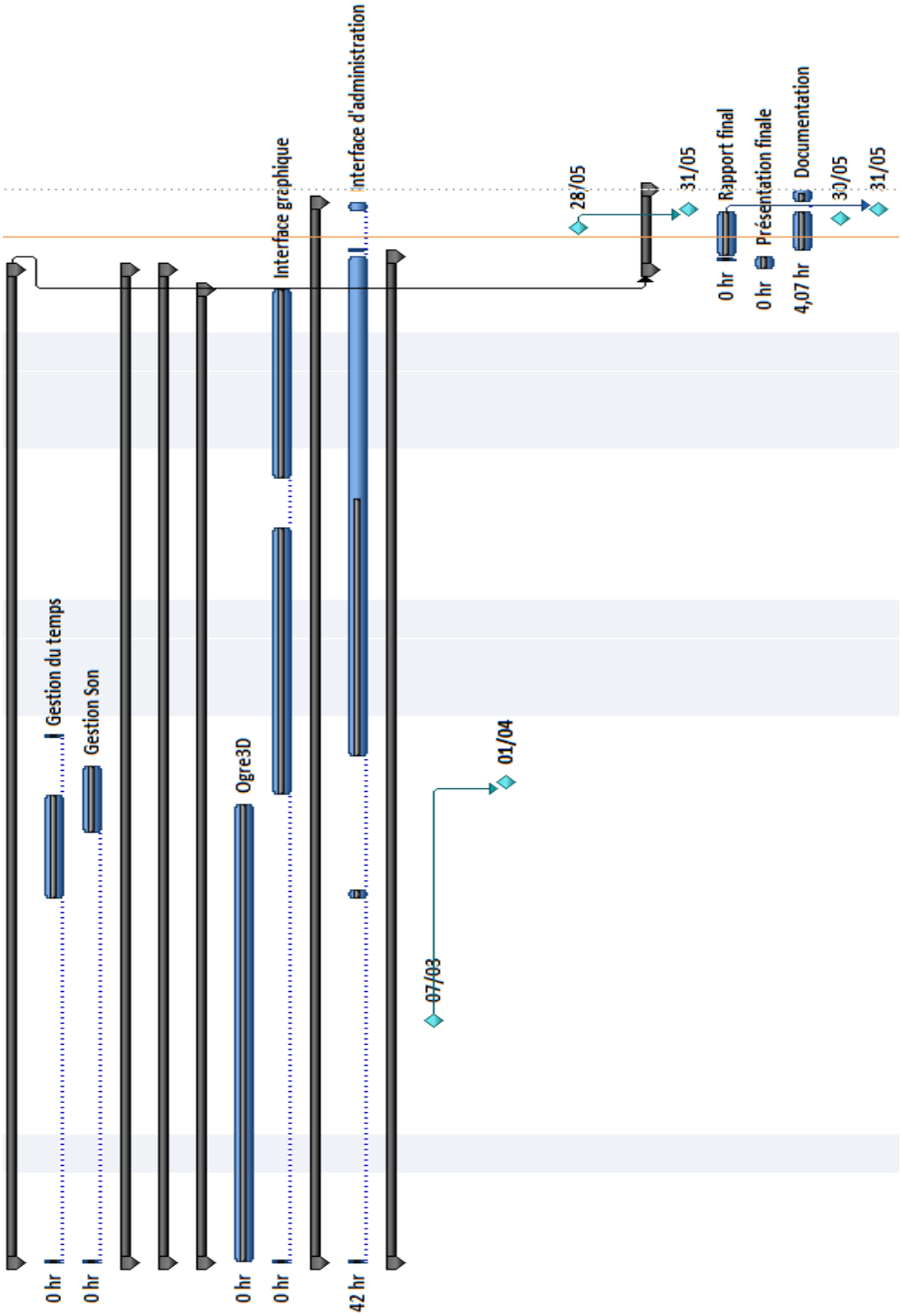

FIGURE 9 – Diagramme de Gantt à la date de livraison du projet

2.2.4 Retards et causes

La plupart des tâches ont pu être menées à terme. Toutefois les tâches suivantes n'ont pas pu être terminées :

Tâche	Avancement à la date de livraison
Interface d'administration	30%
Intégration de Ogre3D dans OpenMASK	30%
Test d'intégration	50%
Recette	10%

Cependant, ces retards proviennent des causes suivantes :
- La **difficulté rencontrée avec l'utilisation d'OpenMASK** : son apprentissage, son utilisation, et sa documentation [8]
- Un des membres du groupe, pour raison de santé a dû **abandonner le projet** à partir de mars.
- Une **tâche supplémentaire**, l'installation, l'intégration et l'utilisation de PhysX [9], cette tâche n'étant tout simplement pas prévue initialement
- Des difficultés rencontrées avec l'**utilisation de VRPN**
- Des **problèmes matériels rencontrés** : une nécessité de changer les nunchuks au profit de versions officiels, une panne des lunettes immersives Vuzix Wrap 920.

Aussi, nous pouvons nous livrer à un petit exercice explicitant la **différence entre le nombre d'heures réelles, et le nombre d'heures prévisionnelles** :

Semestre	**Nombre d'heures prévisionnelles**	**Nombre d'heures réelles**
1^{er} semestre	NC	943
2^{er} semestre	825	1136
Total	-	**2079**

Nous obtenons donc une charge totale de **2079 heures** réelles pour l'ensemble de l'année. D'autre part au second semestre, nous avons une charge 38% plus élevée que celle qui avait été prévue.

2.2.5 Chiffres clés

Pour conclure quant à cette analyse, nous allons donner quelques chiffres particuliers à notre projet :

8. Nous remercions tout particulièrement dans ce cadre nos tuteurs et tous ceux qui ont pu nous aider dans cette tâche difficile.

9. PhysX est un moteur de jeu permettant la détection des collisions

Intitulé	Caractéristique
Nombre de lignes de codes écrites et effectives [10]	9500
Nombre de classes codées et effectives	61
Nombre de lignes de code de test	1000
Nombre de classes de test	8
Nombre de lignes de codes	10500
Nombre total de classes codées	69
Nombre de modules OpenMASK	6
Nombre total d'heures	2079

2.3 Bilan sur les paramètres et la démarche de planification

Un fois l'analyse effectuée de la planification effectuée, il est intéressant de s'interroger sur les paramètres initiaux et notre démarche de gestion de projet. En effet, ceci permettrait de nous indiquer si ces paramètres ont été correctement choisis ou non. Nous nous appuierons une fois de plus sur le rapport de planification.

2.3.1 Bilan sur les paramètres de planification

Les paramètres de la planification étaient, d'une manière générale, bien choisis. Seul un paramètre n'a pu être respecté : le déploiement de l'application n'a pu être effectué qu'en salle μRV et n'a pas pu être réalisé dans la salle Immersia. D'autre part, concernant les risques envisagés, **trois risques, sur les 6 prévus initialement se sont effectivement déroulés** à savoir :

- Un développement nécessitant plus d'heures (OpenMASK)
- Panne, perte de périphérique : une panne des lunettes immersives et le besoin de changer les nunchucks.
- Vol de périphérique : une webcam, appartenant au projet des $5^{èmes}$ année.

Pour chacun des événements ci-dessus, bien que nous ayons appliqué les moyens de préventions associés, nous avons également appliqué la procédure à suivre. Toutefois, **nous n'avions pas prévu qu'un des membres de l'équipe puisse abandonner le projet**.

2.3.2 Bilan sur les méthodes de gestion de projet

Nous nous intéressons aux méthodes de gestion de projet qui ont été prévues et utilisées :

- La méthode RUP [11] (processus unifié) cadre bien avec le rythme sous-jacent au rythme de l'année. En effet, les quatre phases se sont déroulées : analyse, conception, développement et intégration. Dès nos premiers rapports d'analyse, notre projet a été dirigé par les diagrammes de cas d'utilisation.

10. On entend le terme « effectif », le code des les lignes de codes qui sont effectivement exécutées au lancement de l'application
11. Rational Unified Process

– La méthode SCRUM[12] a plus ou moins bien été appliquée. Les différentes réunions, la mise en place d'un système de communication (IRC[13]), le respect des jalons témoignent de son utilisation. Toutefois, elle est parue peu adaptée aux semaines où la charge de travail autre que celle du projet est élevée.

12. La méthode SCRUM est une méthode de développement agile, où les périodes de développement se réalisent en sprints de 20 à 30 jours, qui se terminent par un jalon. Lors de ces jalons, la mise au point d'au moins un produit partiel est prévue.

13. Internet Relay Chat. IRC est un protocole de communication textuelle sur Internet.

3 Perspectives et méthodologie de la reprise du projet

3.1 Travail restant et perspectives de reprise du projet

Quelques fonctionnalités prévues n'ont pas pu être implémentées par faute de temps. Ainsi, la gestion de notre application en mode distribué n'est pas optimale et pourrait être améliorée, notamment le déplacement des utilisateurs qui se fait de façon saccadée et non fluide. Cela devrait être possible de le corriger en développant un objet de simulation OpenMASK qui s'occuperait d'un déplacement fluide.

Une amélioration du support Linux peut aussi être envisagée, telle que la gestion des collisions ou un mode distribué pour Linux. Nous n'avons pas réussi à utiliser le wrapper de NxOgre pour Linux puisqu'il ne dispose pas de support officiel, mais celui-ci pourrait être modifié pour avoir une version Linux compatible. Concernant le mode distribué, nous n'avons pas de version d'OpenMASK pour Linux mais il devrait être possible de le compiler soit-même depuis les dépôts.

Il existe de nombreuses possibilités pour repartir de notre application pour de nouveaux projets. L'interactivité entre les personnages peut être augmentée. Ajouter des pièges est une idée pour y arriver. L'introduction d'un troisième utilisateur dans Block3D, qui utiliserait le joystick inutilisé ne devrait pas trop poser de problèmes, mais cela nécessiterait de remanier les algorithmes d'interaction et de faire un client VRPN pour le joystick.

On peut aussi changer le type de notre application pour, par exemple, en faire un jeu de plateforme utilisant les briques Lego comme éléments de décor. Il faudrait alors repenser la façon de générer les cartes de l'application mais l'ensemble serait assez proche de Block3D.

3.2 Vers de nouveaux horizons

3.2.1 Choix des technologies

Block3D est un mélange de beaucoup de technologies différentes. Il peut servir d'exemple pour l'utilisation combinée du son spatialisé (OpenAL), de la gestion des collisions (PhysX), de la gestion des périphériques (VRPN) et de la gestion de rendu graphique (Ogre). Chacune de ces parties est indépendante et dépend du choix du développeur. Par exemple, si un autre moteur 3D devait être utilisé, notre code ne servira que d'exemple sur la façon de procéder puisque tous les moteurs 3D ont des fonctionnements équivalents. Néanmoins, certaines technologies se sont imposées d'elles-mêmes telles que CMake, qui est utilisé nativement par OpenMASK, Ogre et VRPN. D'autres ont été choisies parce qu'elles faisaient parties des plus matures comme OpenAL ou PhysX, qui sont utilisées dans beaucoup de grands projets commerciaux. Ainsi, même si de futurs projets devaient repartir de zéro, ils devront se demander si partir sur d'autres technologies leurs seraient avantageux.

3.2.2 Reprise de notre travail

Notre modèle est faiblement couplé et a été validé grâce à des tests unitaires. Il peut ainsi être simplement récupéré. Le module pour la récupération des évènements de la Wiimote peut aussi être récupéré. Un effort particulier sur la gestion des dépendances à été réalisé. Ainsi, de nombreux scripts CMake permettent d'importer simplement de nouvelles dépendances dans un projet tel que OpenAL, PhysX ou encore VRPN et ce, que l'on soit sous Linux ou Windows. Enfin, toute la documentation sur la configuration des logiciels et du matériel est destinée aux futurs projets puisqu'elle est sous la forme d'un wiki[2] hébergé par l'INSA. L'idéal est bien entendu que ces différentes parties ne soient pas perdues et quelles soient agrémentées dans le temps. Un manuel d'installation et d'utilisation est également disponible sur ce wiki et sur la forge[1] de l'INSA de Rennes.

3.3 Répartition de la salle

Du mois de septembre jusqu'au mois de février nous avions à gérer deux salles. Nous avions ainsi deux clefs pour la salle de stockage du matériel et une clef pour la salle de projection (salle μRV). Cette disposition n'était pas pratique puisqu'elle impliquait des va-et-vient avec le matériel entre les deux salles dans un couloir séparé par une porte sécurisée. A partir du mois mars, tout le matériel à été transféré dans la salle de projection. Les serrures de ces deux salles ont été inversées et nous avons rendu la clef qui était devenue inutile. Les deux personnes qui se rendaient le plus souvent en salle μRV ont été désignées possesseurs officiels des clefs restantes et il y avait toujours une personne vivant à l'INSA qui avait une clef de la salle (sauf pendant les vacances). Des prêts temporaires étaient parfois effectués si une personne avait besoin d'accéder à la salle. Puisque que nous étions tous dans le même groupe, nous n'avons rencontré aucun problème à nous les échanger et les clefs prêtées étaient retournées le jour d'après dans le pire des cas. Il semble que deux clefs soient l'idéal pour un groupe de projet comme le nôtre. Une rendrait l'accès à la salle difficile et plus rendrait la gestion des clefs difficiles.

Conclusion

Le projet se termine avec une application fonctionnant correctement et respectant la plupart des objectifs au départ du projet. Les deux utilisateurs peuvent évoluer dans le même environnement en s'affrontant dans un labyrinthe en Legos. Le coureur se déplace à l'aide de la Kinect et observe l'environnement sur un vidéoprojecteur. Le constructeur utilise une Wiimote et peut utiliser les lunettes immersives Wrap 920 pour percevoir le monde. Pour ce qui concerne la gestion de la salle, aucun soucis majeur n'est survenu.

Le projet est donc à un stade bien avancé. L'aboutissement du projet n'est pas aussi poussé que ce que l'on aurait pu souhaiter, mais cela peut s'expliquer par les nombreux logiciels auxquels nous devions nous familiariser ainsi que par les problèmes rencontrés. La distribution sur plusieurs machines à l'aide d'OpenMASK est le principal soucis que nous aillons rencontré. Néanmoins, notre application est parfaitement utilisable sans cela.

Concernant la planification, un retard est clairement apparu dans l'avancement des différentes tâches. Cela peut s'expliquer par la complexité d'utilisation d'OpenMASK, l'apparition d'une tâche supplémentaire pour gérer les collisions, les problèmes matériels rencontrés et également par le départ d'un des membres du groupe en mars. Malgré ce retard, nous avons réussi à obtenir une application fonctionnelle.

Enfin, les perspectives de reprise du projet sont grandes. En effet, de nombreux éléments peuvent encore être implantés comme une distribution optimale de notre application, ou encore un troisième utilisateur utilisant le matériel restant de la salle (le joystick notamment). Block3D est également très modulaire et les choix technologiques peuvent être modifiés relativement facilement. La compilation peut ainsi s'effectuer sous Windows et Linux, mais la distribution et la gestion des collisions ne sont présentement pas disponibles sous Linux.

Nous avons donc pu acquérir de nombreuses connaissances tout au long du projet, aussi bien du point de vue de la conception de l'application et de sa planification que du point de vue des technologies utilisées lors de sa réalisation. Ce projet fut riche et intéressant et c'est avec regret, mais fierté, que nous le laisserons aux générations futures.

Bibliographie

Références

[1] Dépôt officiel du projet, disponible sur
https ://forge.insa-rennes.fr/redmine/projects/microrv/

[2] Wiki officiel du projet, disponible sur
http ://micro-rv.insa-rennes.fr/

[3] Projet de l'an passé, disponible sur
http ://l3g0.net

[4] Site officiel d'Ogre3D, disponible sur
http ://www.ogre3d.org/

[5] Forum officiel d'OgreAL, disponible sur
http ://www.ogre3d.org/addonforums/viewforum.php ?f=10

[6] Site officiel d'OpenAL, disponible sur
http ://connect.creativelabs.com/openal

[7] Wiki officiel de NxOgre, disponible sur
http ://www.ogre3d.org/tikiwiki/NxOgre

[8] Lien de téléchargement de Critter, disponible sur
https ://github.com/betajaen/critter/zipball/buggyswires

[9] Site de téléchargement de PhysX, disponible sur
http ://supportcenteronline.com/ics/support/default.asp ?deptID=1949

[10] Site officiel d'OpenNI, disponible sur
http ://openni.org

[11] Site officiel de VRPN, disponible sur
http ://www.cs.unc.edu/Research/vrpn/

[12] Site officiel de FAAST, disponible sur
http ://projects.ict.usc.edu/mxr/faast/

[13] Site officiel d'OpenMASK, disponible sur
http ://www.openmask.org/

[14] MPI, disponible sur
http ://en.wikipedia.org/wiki/Message_Passing_Interface

[15] Site officiel de CppUnit, disponible sur
http ://www.freedesktop.org/wiki/Software/cppunit

[16] Site officiel de CxxTest, disponible sur
http ://cxxtest.com/

[17] Site NVidia, disponible sur
http ://www.nvidia.fr/

[18] Site Vuzix, disponible sur
http ://www.vuzix.com/